

A Vision-based Scheme for Kinematic Model Construction of Re-configurable Modular Robots

Kewei Lin, Juan Rojas, and Yisheng Guan¹.

Abstract—Re-configurable modular robotic (RMR) systems are advantageous for their reconfigurability and versatility. A new modular robot can be built for a specific task by using modules as building blocks. However, constructing a kinematic model for a newly conceived robot requires significant work. Due to the finite size of module-types, models of all module-types can be built individually and stored in a database beforehand. With this *priori* knowledge, the model construction process can be automated by detecting the modules and their corresponding interconnections. Previous literature proposed theoretical frameworks for constructing kinematic models of modular robots, assuming that such information was known *a priori*. While well-devised mechanisms and built-in sensors can be employed to detect these parameters automatically, they significantly complicate the module design and thus are expensive. In this paper, we propose a vision-based method to identify kinematic chains and automatically construct robot models for modular robots. Each module is affixed with augmented reality (AR) tags that are encoded with unique IDs. An image of a modular robot is taken and the detected modules are recognized by querying a database that maintains all module information. The poses of detected modules are used to compute: (i) the connection between modules and (ii) joint angles of joint-modules. Finally, the robot serial-link chain is identified and the kinematic model constructed and visualized. Our experimental results validate the effectiveness of our approach. While implementation with only our RMR is shown, our method can be applied to other RMRs where self-identification is not possible.

I. INTRODUCTION

Modular robotic systems (MRSs) use robotic modules as building blocks to create a variety of kinematic configurations, see Fig. 1. Each module is an independent mechatronic subsystem with a single function. Modules can be of different types. A new kinematic configuration can be specifically selected for a given task and built with modules of different types. This system design principle brings great versatility and flexibility. Throughout this paper, the phrase “kinematic configuration” refers to the robot morphology, whereas the word “configuration” indicates the robot state.

MRSs can be self-reconfigurable robots (SRRs) or not. Unlike SRRs, e.g. Roombot [1] and M-TRAN [2], re-configurable modular robots (RMRs) tend to be industry-oriented and are set-up manually by users for specific tasks [3]–[5]. RMRs reconfiguration is conducted by users, whereas SRRs use complex self-docking mechanisms to ensure reliable mechatronic connections between modules

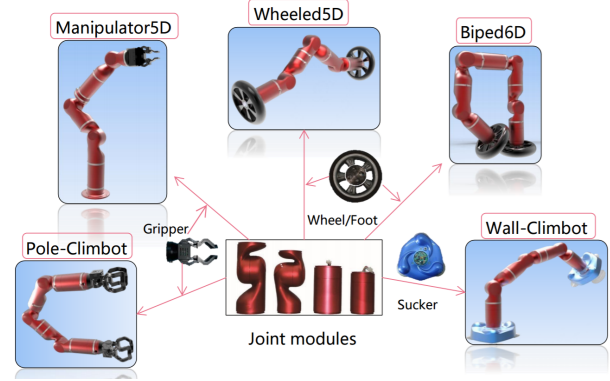


Fig. 1: Different kinematic configurations are possible by combining different robot modules in different ways. Also we classify module types in two categories: joint-modules and tool-modules. Joint-modules build the main body of the robot, while tool-modules act as end-effectors.

and enable self-reconfiguration [6]. RMRs are simpler and of lower-cost compared to SRRs. Nevertheless, they share similar challenges in design and control [7].

To use a RMR for a specific task, one must (i) set up the physical modular robot and (ii) construct the robot model in the software. Physical connection of modules has been well addressed through the use of quick-lock mechanisms in each module. Modular robots are now set up conveniently with little effort. The bottle neck of RMRs application lies in constructing kinematic models for newly conceived modular robots, which takes effort and expertise [4].

When considering kinematic configurations of a RMR system, there is a large number of possible combinations given a finite-set of modules and module-types. It’s cumbersome for RMR designers to derive all possible kinematic models. Consequently, the user needs to derive a kinematic model every time a new modular robot is conceived. The automation of the kinematic model construction is crucial to further lower the barriers for non-experts.

With regard to automatic model generation, Chen *et al.* [4] introduced a theoretical framework for automatic RMR model generation. This work assumed that the robot structure was known *a priori* or specified manually. Rebots, a physics-based simulator for MRS, provides an intuitive way to specify modular robot structures [8]. Given a physical modular robot, a user can observe the module-types and their ordering, and reproduce it in the simulation using drag-and-drop interaction. Despite the intuitive interface, untrained

¹Kewei Lin, Juan Rojas (the corresponding author, Email: juan.rojasn@gdut.edu.cn), and Yisheng Guan are with the Biomimetic and Intelligent Robotics Lab (BIRL), School of Electro-Mechanical Engineering, Guangdong University of Technology, Guangzhou, China, 510006.

users may mistake the module-types or their ordering.

The goal then is to automate both the identification of the robot structure as well as the kinematic model generation. In doing so, model errors will be minimized and barriers-to-use will be lowered. For developing plug-and-play RMRs, modules should be able to self-identify: (i) *the order* in which they are connected and (ii) *how* adjacent modules are connected. There are in fact, self-reconfigurable modular robots that are able to perform self-identification of their kinematic structures. They do so, through built-in sensors that detect the adjacent modules and the connection between modules. M-TRAN [2], uses IR communication between neighboring modules to learn connection information. The hardware solution (i.e. the use of additional built-in sensors) increases the design complexity of the docking interface, increases costs, and yet the information query is done infrequently. For industrial-oriented RMRs, reconfiguration happens only occasionally (only when new tasks are given), so the inclusion of additional built-in sensors is less preferred.

In this paper we study whether less costly alternatives exist to autonomously identify the robot kinematic configuration and subsequently generate the kinematic model of the system. To this end, we contribute a vision-based kinematic-configuration identification system and an accompanying kinematic model generation scheme for RMRs. Our work places one or more markers (QR codes, laser engravings, etc) in each module. Each marker has a unique ID. A camera takes a picture of the modular robot. Markers are detected by the marker tracking method and their poses are used to estimate the poses of the modules. By querying the module database, other information of the corresponding modules is also available. Then with this set of detected modules and their pose estimates, the kinematic structure grows by iteratively (i) finding the connected module for each module and (ii) computing their connection parameters. After identifying the modules and their connections, the robot kinematic chain is constructed. Moreover, joint-modules' joint angles are also computed to better visualize the generated robot model. The effectiveness of our approach is demonstrated by the correct model generation of multiple robots with different kinematic configurations or a robot at different configurations.

This paper is organized as follows: In Sec. II, the general identification framework for tree-type RMRs (chain type as a special case) is introduced. Sec. III and Sec. IV detail the implementation of our method specifically for our RMR system. In Sec. III, terminology and representation of RMR are introduced. In Sec. IV kinematic chain identification and model construction are described. In Sec. V experiments are shown. In Sec. VI discussion of the topic is presented.

II. GENERAL FRAMEWORK OF KINEMATIC CHAIN IDENTIFICATION

In our method, a camera is used to take images of a newly built modular robot (see Fig. 2). AR-tag-tracking methods (e.g. `ar_track_alvar` [9]) can be employed to detect marker tags on a modular robot and estimate their poses.

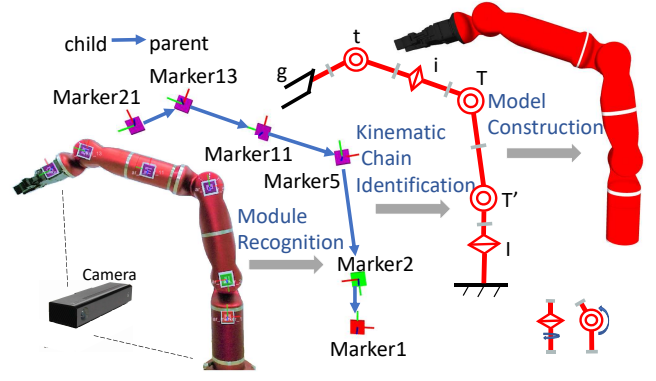


Fig. 2: Each module is affixed with AR tags. The camera takes an image of the newly built modular robot, a manipulator in this case, for identification of the kinematic structure. Then the robot model is constructed and visualized.

A module database maintains data of all module-types and information of all fabricated modules. Querying the module database with marker IDs retrieves information of the corresponding modules, including the module-types and communication IDs on the field-bus.

Then the procedure of kinematic structure identification begins to find all parent-child relations among detected modules. In a pair of connected modules, the parent module is the module that is closer to the base of the chain, and the child is the remaining one. A tree-like kinematic structure has one or more modules as its end-effectors, which are called tool-modules, in order to perform useful operations.

An iterative algorithm will identify parent-child relations to build the kinematic chain (see Fig. 3). The algorithm starts by selecting a tool-module as a child and finding its parent. For each succeeding cycle, parent modules become child modules. The chain grows until no further parents are found. This is considered a kinematic branch-chain. In cases, where other branch-chains exist, those would be processed as well resulting in an overall tree-structure for the modular robot. For chain-like RMR systems, since there is only one branch chain, the algorithm computes the chain in one pass.

As our approach deals with tree-type structures, the parent-searching method avoids multiple results that appears in a child-searching method, because in a tree-structure a module might have multiple children.

One way to search for a parent module of a child is to exploit the geometric constraints of all module-types, as we will detail in Sec. IV.

Another alternative is to formulate the problem as an optimization problem. In doing so, a more generalized framework for different RMRs can be established. What's more, with a well-devised cost function, all variables needed to extend the kinematic structure can be found simultaneously, including the parent module, the connection and the state of the parent. The dimension of optimization space does not grow as the degrees of freedom of a modular robot increase, because each time only variables of one module is optimized.

For two modules m_1 and m_2 , relevant definitions are listed

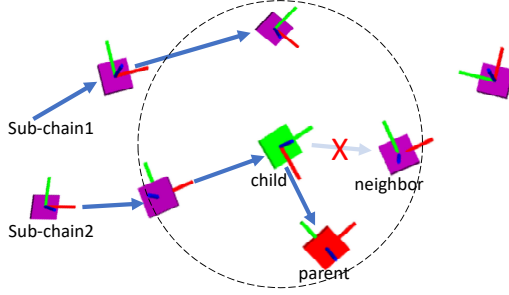


Fig. 3: Kinematic sub-structures extend themselves by iteratively finding a parent module among detected modules for a child. In the end, all sub-structures form an overall kinematic structure of a modular robot.

in Table I.

TABLE I: NOMENCLATURE

M	a set that contains all modules types
t_{m_1}	module-type of m_1 , query from module database
H_{m_1}	detected pose of m_1 as a homogeneous matrix
$O_{m_1}xyz$	coordinate frame of H_{m_1}
θ_{m_1}	module state, e.g. a joint angle for a joint-module
c_{m_1, m_2}	connection variable between parent m_1 and child m_2
N_{m_1}	neighboring modules around m_1

For most RMR systems, only finite possible connection approaches between two modules are supported: $c_{m_1, m_2} \in \{c_1, c_2, \dots, c_n\}$.

Given a child module, denoted as c , the algorithm tries to find out its parent p . To do so, first, neighbors around c , N_c , are found. The distance between a neighbor n and c must not exceed the maximum distance between two connected modules of any module-types and therefore satisfies Eqtn. 1:

$$\|O_n \vec{O}_c\| \leq \max(\|O_p \vec{O}_c\|) + \epsilon_1, \forall n \in N_c. \quad (1)$$

where ϵ_1 is an error threshold.

For each neighbor $n \in N_c$, the transformation from n to c is defined as a homogeneous matrix $T_c^n = H_n^{-1} H_c$. If n is the parent of c , i.e. $p = n$, a module state θ_n and a connection variable $c_{n, c}$ that transforms the pose of c to that of n can be found. This transformation is defined as

$$T(c_{n, c}, \theta_c, t_c, \theta_n, t_n) = T_1(\theta_c, t_c) T_2(c_{n, c}) T_3(\theta_n, t_n) \quad (2)$$

where T_1, T_2 and T_3 denote respectively the transformations relevant to child's state, connection between parent and child, and parent's state.

In other words, if $p = n$, there is a θ_n and a $c_{n, c}$ that allows T in Eqtn. 2 to theoretically equal T_c^n . A metrics F is devised for measuring the distance between T and T_c^n :

$$F(T, T_c^n) = \|W \circ (T - T_c^n)\|_2, \quad (3)$$

$$W = \begin{bmatrix} w_o & w_o & w_o & w_t \\ w_o & w_o & w_o & w_t \\ w_o & w_o & w_o & w_t \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

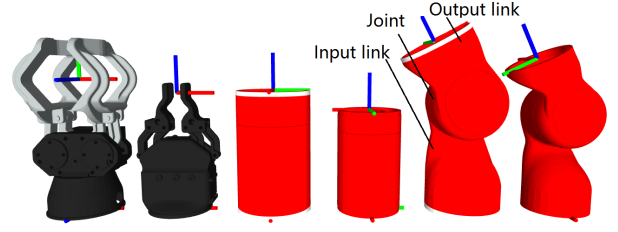


Fig. 4: From left-to-right, we have two different types of tool-modules that serve as grippers, followed by 4 joint-module types. The first two are simple cylinders that allow for revolute rotations, the last two are composed of two revolute joints and serve as T-connectors. They are denoted as G, g, I, i, T and t .

where \circ is element-wise product and a weighting matrix W that adjusts weights for translational and orientation differences.

The parent-searching procedure now can be expressed in the form of an optimization problem:

$$\begin{aligned} \min : & F(T, T_c^n) \\ \text{variables} : & n, c_{n, c}, \theta_n \\ \text{s.t. } & n \in N_c; \\ & c_{n, c} \in \{c_1, c_2, \dots, c_n\}; \\ & \theta_n \in [\min(\theta_{t_c}), \max(\theta_{t_c})]. \end{aligned} \quad (4)$$

If $\min(F(T, T_c^n))$ is within a certain threshold, the parent $p = \arg \min_{n \in N_c} F(T, T_c^n)$ is found, as well as the connection variable with its child and the parent state. This parent-searching procedure extends the kinematic chain to next module. And one module by another, the kinematic chain can be completed.

III. TERMINOLOGY AND REPRESENTATION OF RE-CONFIGURABLE MODULAR ROBOTS

The following two sections will detail the implementation of our method specifically for our RMR system in [3]. This section introduces the representation of modules and modular robots in [3].

A. Re-configurable Modular Robotic System

Our system and its design methodology are well presented in [3]. Fig. 1 shows five robotic systems built with these modules for different applications [3], [10], [11].

For joint-modules, upper-case letters T and I indicate I-typed and T-typed joint-modules whose rotation axes are collinear with and perpendicular to the link axes, respectively. As for tool-modules, we use G, W and S to represent respectively the gripper, wheel and suction modules. Fig. 4 shows some of these modules.

Smaller-sized modules are also developed. Using them instead of larger ones near the end of a serial modular robot help ease the torque requirement of its base module. These modules are denoted with lower-case letters such as t, i and g , which indicate the smaller version of T, I and G .

Besides joint-modules and tool-modules, there are accessory modules. Link modules are used to extend the distance between two adjacent joint axes and are denoted as L or l . Adapter linkage modules, denoted as A , are used to connect larger modules with smaller modules.

Most of these modules can be installed in an inverted way in a kinematic chain, i.e. the output link is closer to the base. A single quote after the module-type letter indicates this type of installation, e.g. T' .

B. Representation of Basic Modules

Each module-type is stored in the module database as a Xacro format file that contains kinematic and dynamic parameters, as well as transmission and motor specifications. These Xacro files are named with the corresponding module-names.

Since the joint-modules can be installed into a kinematic chain in an upside-down way, the terms *input* and *output* are adopted only in the sense of where the driving motors are placed. To address the problem of inverted installation of modules, Xacro files of inverted modules are also created.

The usage of Xacro files incorporates other useful tools from Robot Operating System (ROS) such as the general inverse kinematics solver TRACK-IK [12].

C. Representation of Modular robots

With a database that stores Xacro files of all module-types, we can develop a high-level abstraction of kinematic chains of modular robots by specifying the following information:

- 1) constituent modules and their types,
- 2) the order of each module in the kinematic chain,
- 3) connectivity between a pair of adjacent modules (connection angle c).

Besides, in order to better visualize the robot, we will also estimate the robot state: 4) joint angles of joint-modules.

A text-based method is employed to implement this abstraction for our serial modular robotic system. The modular manipulator in Fig. 1 is described with a string " $I\text{-}T0\text{-}T0\text{-}A\text{-}i0\text{-}t180\text{-}g90$ ". And for the biped tree/truss climbing robot Pole-Climbot in Fig. 1, it is " $G'\text{-}I0\text{-}T'0\text{-}L0\text{-}T'90\text{-}T180\text{-}I'0\text{-}G0$ ". However, it is also appropriate to describe it with " $G'\text{-}I0\text{-}T'0\text{-}T'180\text{-}L'(-90)\text{-}T0\text{-}I'0\text{-}G0$ " because of its bipedal locomotion behavior. These strings are organized in the order from the bases of kinematic chains to the ends. The hyphens are delimiters between adjacent substrings. Each substring contains the module-type and the connection angle c in degrees. The connection angle describes how a module is connected to its parent module. Due to four pin holes on the connection interface of each module, there are only four possible connection angles: $c \in \{-90^\circ, 0^\circ, 90^\circ, 180^\circ\}$.

Once the above high-level parameters are specified, a modular robot can be represented with an overall Xacro file, which incorporates macros of the constituent modules according to these parameters.

While straightforward enough for MRS designers, the text-based method still requires users to manually specify the robots structure, which is error-prone for users.

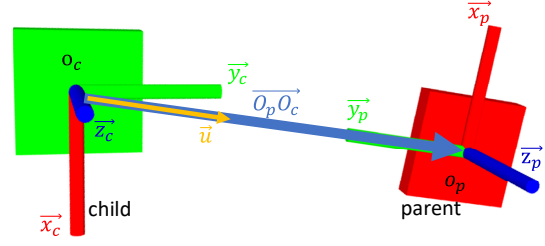


Fig. 5: A green rectangle represents the master marker of the child module c while the red one represents that of the parent p . The frame O_cxyz represents the input link's pose of c while O_pxyz represents that of p . \vec{u} is the unit vector of O_pO_c .

IV. KINEMATIC CHAIN IDENTIFICATION

To identify the aforementioned high-level parameters, we propose a vision-based method. An open-source project called `ar_track_alvar` [9] is employed to track AR tags on modules.

A. Placement of Tags

Every module is affixed with marker tags encoded with unique IDs. The markers are placed on the surface of each module such that at least one of them can be captured by the camera while we seek to keep the number of markers to minimum. Markers on the same rigid body are called a "bundle". The usage of multi-tags bundle allows to estimate the pose of a multi-sided object, even when some of the tags cannot be seen. One marker in each bundle is designated as the master marker. The poses of the detected markers are transformed into the pose of the master to represent that of the object.

Each I-type module has two bundles, one for the output link and the other for the input link. This allows computing the joint angle θ using the difference between the poses of two linkages $T_{out}^{in} = H_{in}^{-1} H_{out} = Rot(y_{in}, \theta) Trans(y_{in}, dis)$, where dis represents the translational distance between two poses.

For other module-types, only input links are attached with tags in order to reduce the number of tags. For a T-type module, the joint angle can be computed by comparing the pose of its input link with that of the adjacent modules.

A virtual marker, locating at the center of each module with its Y-axis collinear with the axis of input link, is designated as the master marker of the module. For each T-type module, the Z-axis of the virtual master marker is collinear with the joint axis. Using these virtual master markers to represent poses of modules are convenient for calculating high-level parameters. Now the Y-axis or its reverse of a module is pointing to the parent module, with the only exceptions of inverted T-type modules T' , t' . This allows the use of a geometric method instead of the optimization method for our system.

B. Module Identification Using Markers

The ID of each master marker on a module represents the serial number of this module, which is unique and logged in a database that maintains all product information of the fabricated modules. The product information database also manages all the module Xacro files.

Once the markers are detected, they are transformed to the corresponding virtual master markers. By querying the module database, the module-type, ID number on communication bus and other information of each marker are available for further computation.

C. Geometry Constraints

Our RMR system shows an obvious geometric characteristic: the y-axis of a module's pose usually points to or opposes to its adjacent module, with exceptions when T-type modules get involved. Therefore, a geometry-based method will be more convenient for our implementation.

The markers tracking and database querying procedures return the poses of constituent modules of a modular robot, which must satisfy a series of geometry constraints. Some useful constraints are deduced as below.

Here define some symbols for the convenience of discussion. Modules installed in an upright way in the kinematic chain belong to the set $m = \{T, t, I, i, G, g, W, S, L, l, A\}$, while the inverted ones belong to $m' = \{T', t', I', i', G', g', W', S', L', l', A'\}$. A complete set of all module-types is defined as $M = m + m'$.

Consider a pair of connected modules, parent module $p \in M$ and its child module $c \in M$, geometric constraints between them can be organized as a constraint matrix CST in Eqtn. 5 according to their module-types. Each element in CST , $cst_{p,c}$ indicates the constraint set that corresponds to the parent and child module-types.

$$CST = \begin{bmatrix} cst_{T,T} & cst_{T,t} & \cdots & cst_{T,i'} \\ cst_{t,T} & cst_{t,t} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ cst_{i',T} & \cdots & \cdots & cst_{i',i'} \end{bmatrix}. \quad (5)$$

As illustrated in Fig. 5, the frame O_pxyz represents the input link's pose of the parent module while O_cxyz represents that of the child. \vec{u} is the unit vector of O_pO_c .

For practical implementation, only some handy constraints, denoted as cst , are used and they are listed as

follows:

$$\begin{aligned} cst1 : & \left\| O_p \vec{O}_c \right\| \leq \max(\left\| O_p \vec{O}_c \right\|_{p,c \in M}) + \epsilon_1 . \\ cst2 : & |\vec{y}_p \cdot \vec{u}| \geq 1 - \epsilon_2 . \\ cst3 : & |\vec{y}_c \cdot \vec{u}| \geq 1 - \epsilon_2 . \\ cst4 : & \vec{y}_p \cdot \vec{u} \geq 0 . \\ cst5 : & \vec{y}_p \cdot \vec{u} < 0 . \\ cst6 : & \vec{y}_c \cdot \vec{u} \geq 0 . \\ cst7 : & \vec{y}_c \cdot \vec{u} > 0 . \\ cst8 : & p \in m . \\ cst9 : & p \in m' . \\ cst10 : & c \in m . \\ cst11 : & c \in m' . \end{aligned} \quad (6)$$

where error thresholds are defined as: $\epsilon_1 \ll \max(\left\| O_p \vec{O}_c \right\|_{p,c \in M})$, $\epsilon_2 \ll 1$. These constraints are summed up in Table II for short.

TABLE II: Geometric constraints of a pair of connected modules.

$child \setminus parent$		$p \notin \{T, t\}$	$p \in \{T, t\}$		
			if $cst6$	if $cst7$	
$c \notin \{T', t'\}$	if $cst4$	$cst1, 2, 3, 8$	$cst1, 10$	$cst1, 11$	
	if $cst5$	$cst1, 2, 3, 9$			
$c \in \{T', t'\}$	if c_4	$cst1, 2, 8$	$cst1$		
	if $cst5$	$cst1, 2, 9$			

By looking up the above table, the parent module can be selected among the neighbors. Note that though Table II is an incomplete table for CST , e.g. $\{cst1, cst2, cst3, cst8\} \in cst_{t,g'}|_{c_4}$, it's useful for finding parent-child relations in a kinematic chain.

Once the parent module is found, the relative pose T_c^p is exploited for calculating the connection angle $c_{p,c}$ and joint angle of parent θ_p . For connection angle $c_{p,c} \in \{-90^\circ, 0^\circ, 90^\circ, 180^\circ\}$:

$$\begin{aligned} c_{p,c} &= discretize(\hat{c}_{p,c}) ; \\ \hat{c}_{p,c} &= \begin{cases} \arccos(\vec{z}_p \cdot \vec{z}_c) & , \text{if } \vec{z}_p \times \vec{z}_c \cdot \vec{u} \geq 0 \\ -\arccos(\vec{z}_p \cdot \vec{z}_c) & , \text{if others} \end{cases} \quad (7) \end{aligned}$$

D. Kinematic Chain Construction

After setting up the physical modular robot, the users launch the robot identification program to start procedures illustrated in Fig. 6.

By querying the module database, the algorithm examines all detected markers for their validity and, if valid, retrieves their module information. False positives are eliminated while valid markers remain and are exploited to build the kinematic chain.

Tool-modules are used as the end-effectors. The procedure of kinematic chain identification starts from the first detected tool-modules and grows along the chain to the base module. In the biped climbing robot, two tool-modules are used, either of which can be regarded as the end of the kinematic chain.

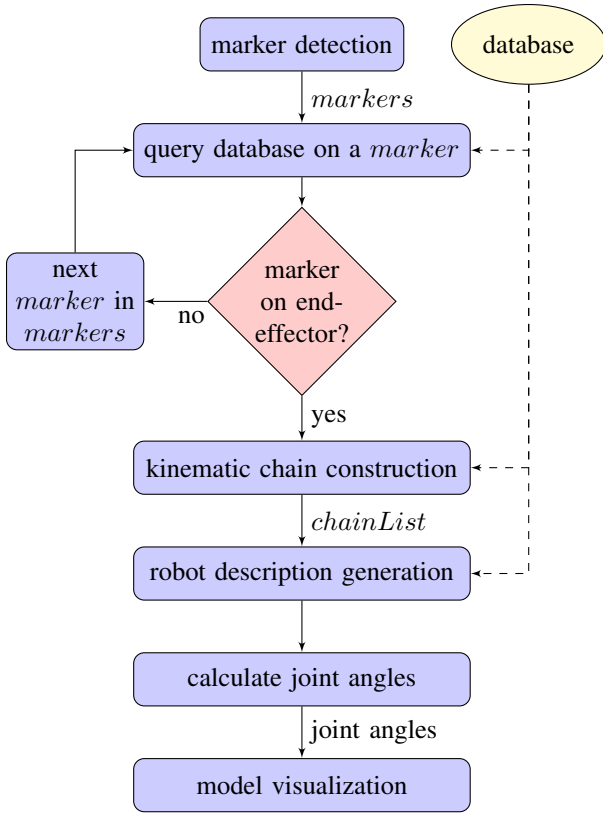


Fig. 6: Flowchart of the modular robot identification.

Once the end module is found, its pose is used for finding its parent module and calculating the connection parameter according to the geometric constraints in Table II. This process continues iteratively until all identified modules find their place in the kinematic chain. A recursive algorithm, see Line 14 in Alg. 1, is designed for the process. Note that *chainList* in Alg. 1 returns the kinematic chain in the order from base to end.

After identifying the kinematic chain, the joint angles are computed for robot model visualization, which allows the users to intuitively confirm the result. Robot specification files, as well as files for control, are also generated. The generated files are essential for our ROS-based control software, which aims to provide a programming framework for modular robots. Our work on robot model identification paves the path towards the goal of kinematic-configuration-agnostic programming for modular robots.

V. EXPERIMENTS

In our method, only a camera is needed. To be specific, a Kinect is used in Fig. 7, but any common camera can be used. Currently AR tags are used, however we envision our modules being marked through laser engravings during fabrication.

Four tests were carried out, as illustrated in Fig. 8. Our algorithm successfully identified the kinematic chains and computed the joint angles. In each sub-figure, the left part shows an image of a modular robot, taken by the camera,

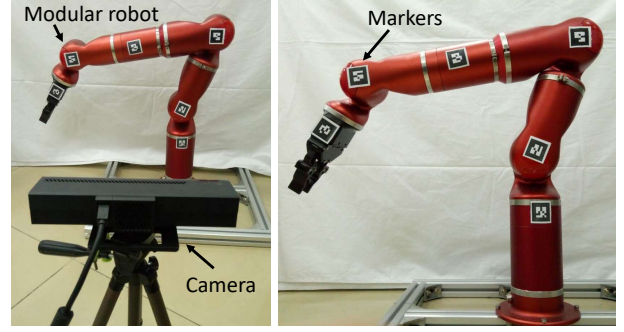


Fig. 7: Experimental set-up for modular robot identification. Only a camera is needed in our method.

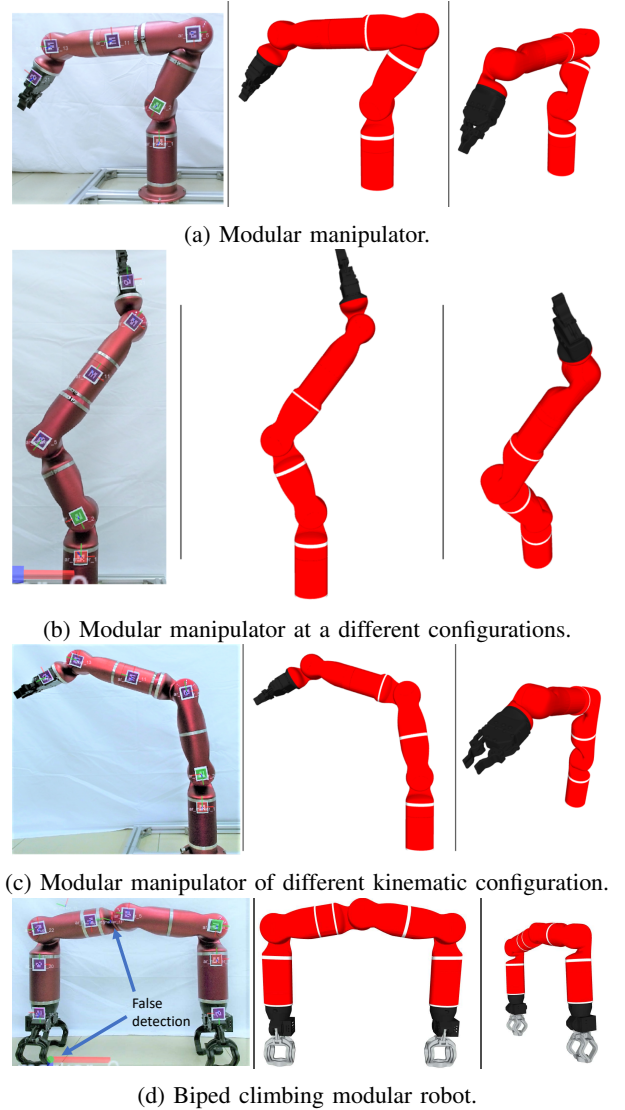


Fig. 8: Three modular robots were tested. The first two experiments used the same robot but at different configurations. Despite the existence of false marker detection, the algorithm could identify the kinematic chains and compute the joint angles for visualization.

Algorithm 1 Kinematic chain construction

```
1: Initialize chainList
2: childMarker  $\leftarrow$  markerOnEndEffector
3: function BUILDCHAIN(childMarker, markers)
4:   if markers is empty then
5:     return
6:   end if
7:   for all marker  $\in$  markers do
8:     if marker is parent of childMarker then
9:       Find connect angle  $c_{p,e}$ , install direction  $d$ .
10:      Remove marker from markers.
11:      break for loop
12:    end if
13:  end for
14:  BuildChain(marker, markers)  $\triangleright$  recursion
15:  Append [marker,  $c_{p,e}$ ,  $d$ ] to chainList.
16: end function
```

while the middle and the right columns illustrate the resulting model, which are constructed on-the-fly.

The first two experiments, shown in Fig. 8a and Fig. 8b, used the same modular manipulator, but at two different configurations. They both shared a same kinematic chain: $I-T^0O-T^0O-AO-tO-iO-gO$. The third experiment identified another modular manipulator with a kinematic chain $I-T^0O-TO-AO-iO-tO-gO$. The robot in Fig. 8d is not a manipulator, but a bipedal climbing robot: $G'-IO-T^0O-LO-T^0O-OT-OI^0O-GO$, showing that the proposed method can be applied to bi-directional kinematic chains such biped robots.

Though kinematic chains were correctly identified, joint angle estimations were not precise due to the errors of tag placement on the module and the detection errors. Some false detections appeared in Fig. 8b and Fig. 8d. They were eliminated as they failed to match the information stored in the module database, or couldn't find any parent or child.

VI. DISCUSSION AND CONCLUSION

In this paper we showed that kinematic-configuration identification and automatic kinematic model generation can be done on the fly by using a vision-based scheme and a geometric mathematical model of serial-chain systems.

Reconfigurability brings great advantages and also challenges, comparing to robots with fix-morphology. The software of RMR systems is supposed to support features corresponding to the reconfigurability. The rapid development of robot middlewares provides useful tools for implementation of software for RMRs. Xacro files, widely used in ROS, can be used to represent different types of modules as XML macros. Generic kinematic solvers such as TRACK-IK and IKFast, with properly specified kinematic structures, provide kinematic solutions for robots. These can greatly simplify the painstaking construction process of kinematic models, assuming that the kinematic chain is known or manually specified.

To identify the kinematic structures of modular robots allows full automation of the model construction process. We

proposed a vision-based scheme for kinematic configuration identification and model construction of RMR in this paper. Each module is affixed with AR tags, which can be laser-engraved on the module shell during fabrication. A camera is used for tracking these markers, whose poses and IDs are used for computing the kinematic chain and joint angles. Finally, the constructed models are visualized for validation. All the above procedures are done on-the-fly. Comparing to solutions that use built-in sensors or other mechatronic device in each module, our method is of lower-cost and can be applied to other modular robotic systems in which self-identification is not possible.

In the further, in order to thoroughly address the occlusion problem, a hand-held camera can be employed to scan a robot while a multi-view geometry algorithm keeps tracking all marker poses. What's more, our ultimate goal is a system that allows kinematic-configuration-agnostic programming for RMR systems, i.e. programming is not specifically for a particular modular robot but for all possible ones.

VII. ACKNOWLEDGMENTS

This work is supported by the grants: "Major Project of the Guangdong Province Department for Science and Technology (2014B090919002), (2016B0911006)."

REFERENCES

- [1] A. Sprowitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. J. Ijspeert, "Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1016–1033, 2014.
- [2] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of m-tran iii modular robotic system," *The International Journal of Robotics Research*, vol. 27, pp. 373–386, 2008.
- [3] Y. Guan, L. Jiang, and X. Zhang, "Development of novel robots with modular methodology," in *Ieee/rsj International Conference on Intelligent Robots and Systems*, 2009, pp. 2385–2390.
- [4] I. M. Chen, H. Y. Song, G. Chen, and G. Yang, "Kernel for modular robot applications: Automatic modeling techniques," *International Journal of Robotics Research*, vol. 18, no. 2, pp. 225–242, 1999.
- [5] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, and G. Grunwald, "The kuka-dlr lightweight robot arm - a new reference platform for robotics research and manufacturing," in *Isr/robotik 2010, Proceedings for the Joint Conference of Isr*, 2010, pp. 1–8.
- [6] J. Baca, S. G. M. Hossain, P. Dasgupta, C. A. Nelson, and A. Dutta, "Modred: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration," *Robotics & Autonomous Systems*, vol. 62, no. 7, pp. 1002–1015, 2014.
- [7] M. Yim, W. M. Shen, B. Salemi, and D. Rus, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.
- [8] T. Collins and W. M. Shen, "Rebots: A drag-and-drop high-performance simulator for modular and self-reconfigurable robots." [Online]. Available: <http://www.isi.edu/publications/trpublic/pdfs/isi-tr-714.pdf>
- [9] S. Niekum, "ar_track_alvar - github repository," 2012, [Online; accessed 9-February-2017]. [Online]. Available: https://github.com/sniekum/ar_track_alvar
- [10] Y. Guan, X. Shi, X. Zhou, X. Zhang, and H. Zhang, "A novel mobile robot capable of changing its wheel distance and body configuration," in *International Conference on Robotics and Biomimetics*, 2009, pp. 806–811.
- [11] Y. Guan, H. Zhu, W. Wu, X. Zhou, L. Jiang, C. Cai, L. Zhang, and H. Zhang, "A modular biped wall-climbing robot with high mobility and manipulating function," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1787–1798, 2013.

- [12] P. Beeson and B. Ames, “Trac-ik: An open-source library for improved solving of generic inverse kinematics,” in *IEEE RAS Humanoids Conference*, 2015, pp. 928–935.